Energy-Aware Microprocessor Synchronization: Transactional Memory vs. Locks

Tali Moreshet **R. Iris Bahar**

Division of Engineering

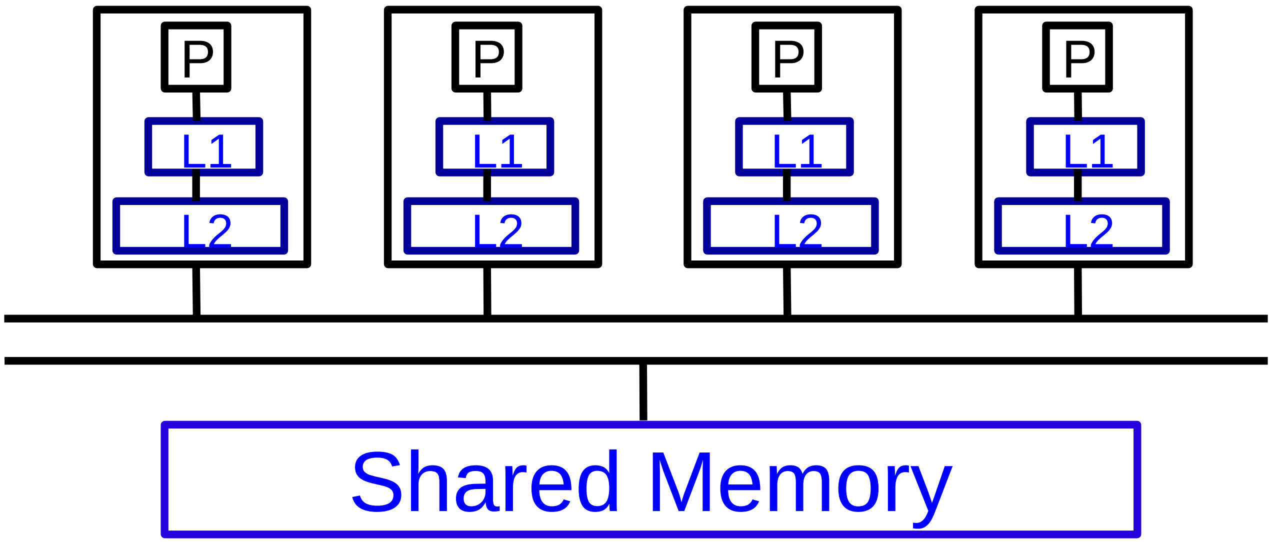
Brown University

Maurice Herlihy

Department of Computer Science

Brown University

Shared Memory Architecture



- Atomic memory access

  Increment variable in address A

Load (R1, A)
Add (R1, R1, 1)
Store (A, R1)

Synchronization of Accesses to Shared Memory

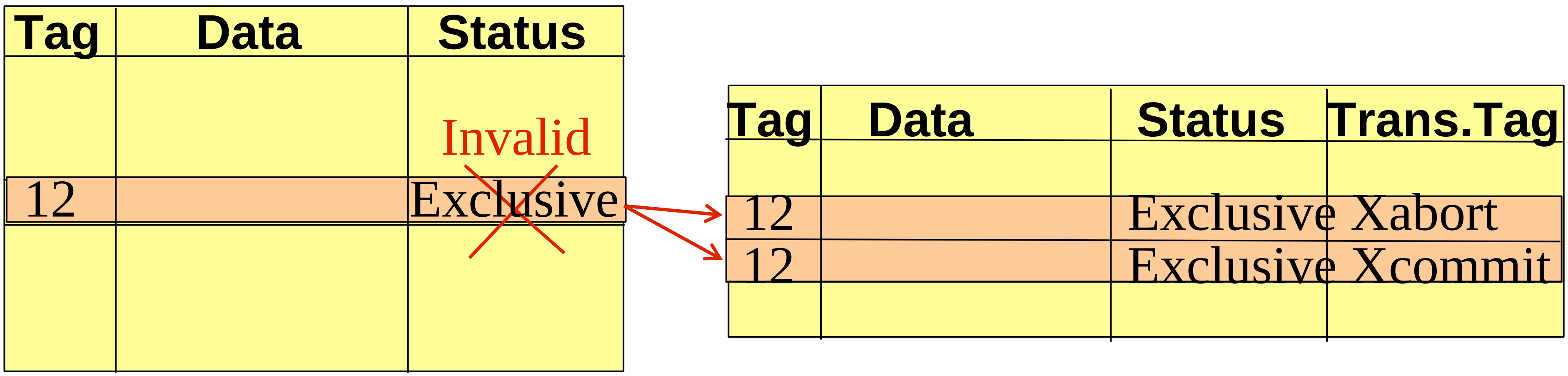
Lock

- Represented by field in memory
- Repetitive accesses until free
- Coarse/Fine-grain
- Disadvantages:
  - High contention
  - Low throughput
  - High energy consumption

Transaction

- Lock-free execution
- Speculative, optimistic
- Ease of programming
- Disadvantages:
  - Requires HW support
  - Roll-back and reissue if conflict detected (wasted cycles and energy)

During a Transaction



- Lookup in both DL1 and transactional cache
- If the line is found in DL1,
  move it to transactional cache
- If a miss, bring from L2 to transactional cache

Considerations

- In the past designers only considered ease of programming and throughput
- Synchronization has a cost in terms of throughput and energy
- We take a first look at tradeoffs for

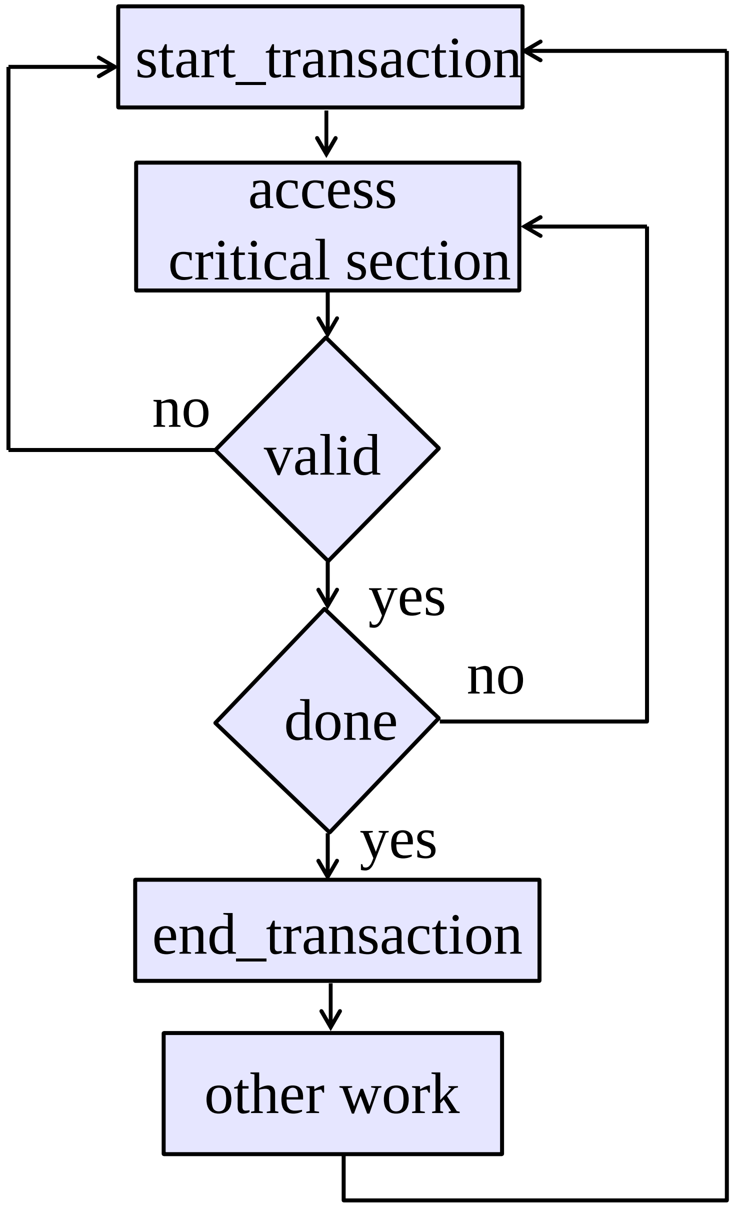
  Ease of programming
  Throughput
  Energy

Energy Consumption per Access

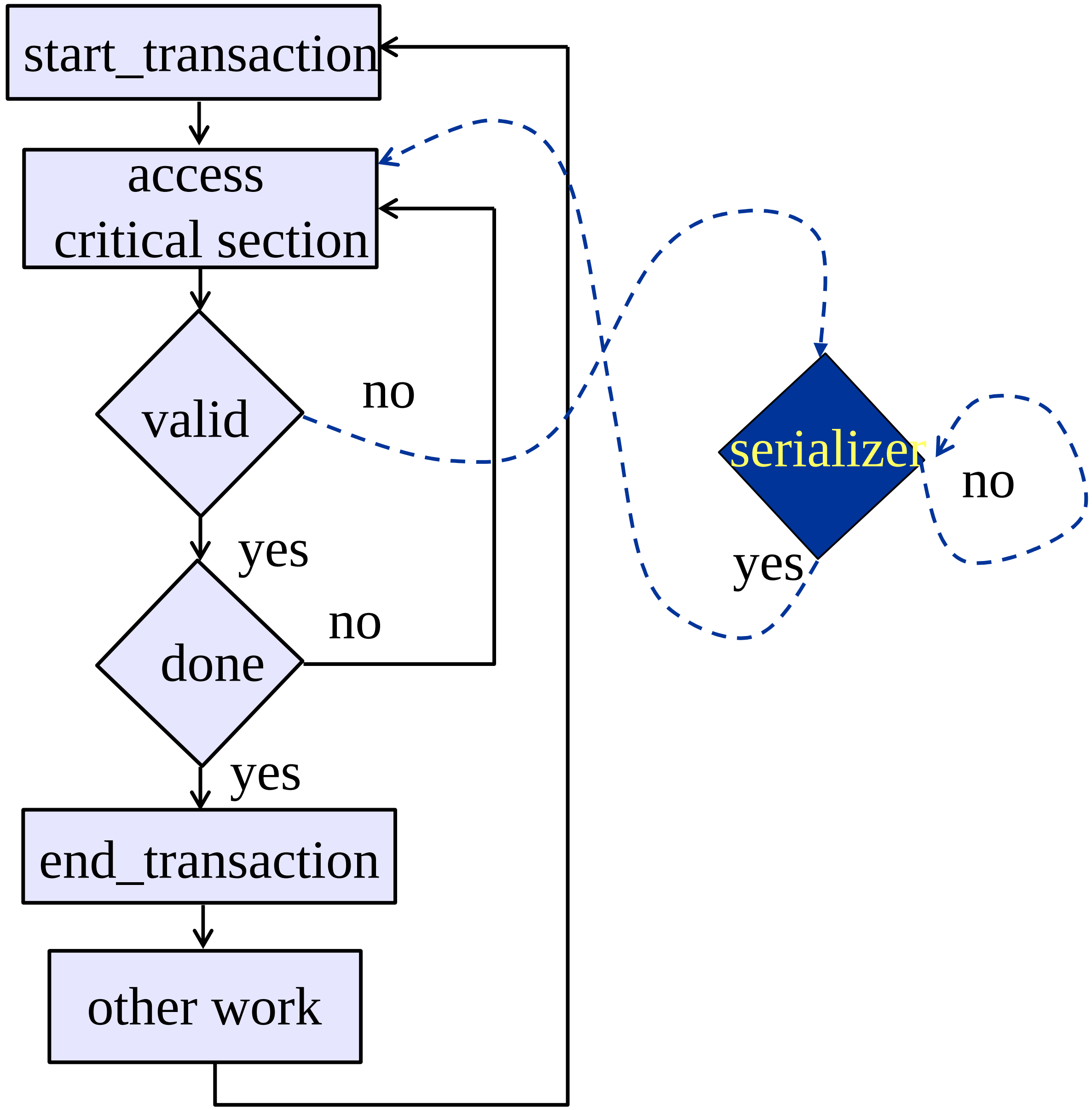
L1Data Cache	8KB 4-way; 32B line; 3 cycle latency	0.47nJ
Transactional Cache	64-entry; fully associative	0.12nJ
L2 Cache	128KB 4-way; 32B line; 10 cycle latency	0.9nJ
Shared Memory	256MB; 64-bit bus; 200 cycle latency;	33nJ

Sources: Micron SDRAM power calculator
CACTI
Private industrial communication

Standard Transactions



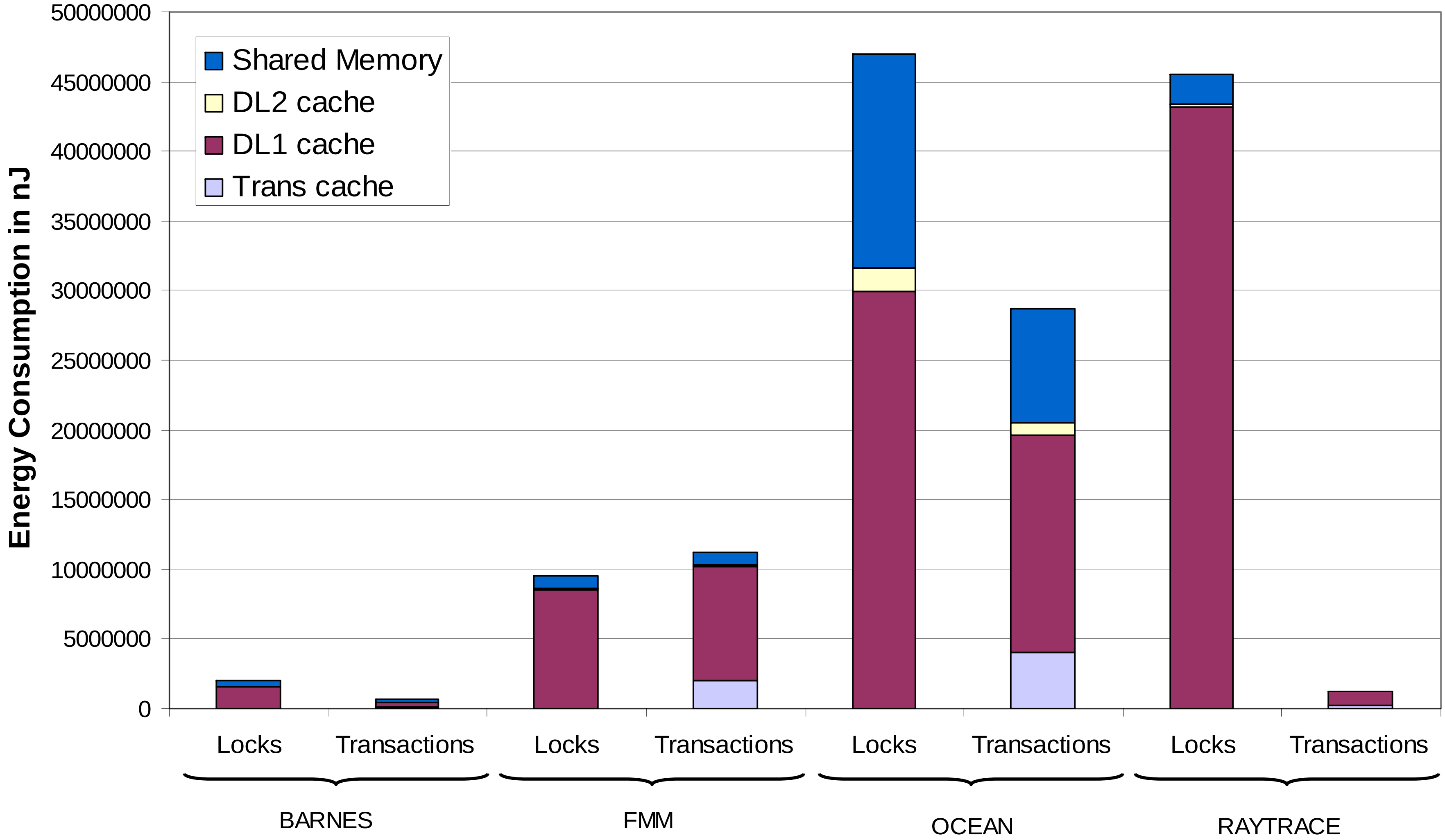
Standard Transactions



Serializer

- Only impacts conflicting transactions
- Small overhead in hardware
- Reduce useless execution
- Reduce energy consumption
- Potentially negative impact on throughput

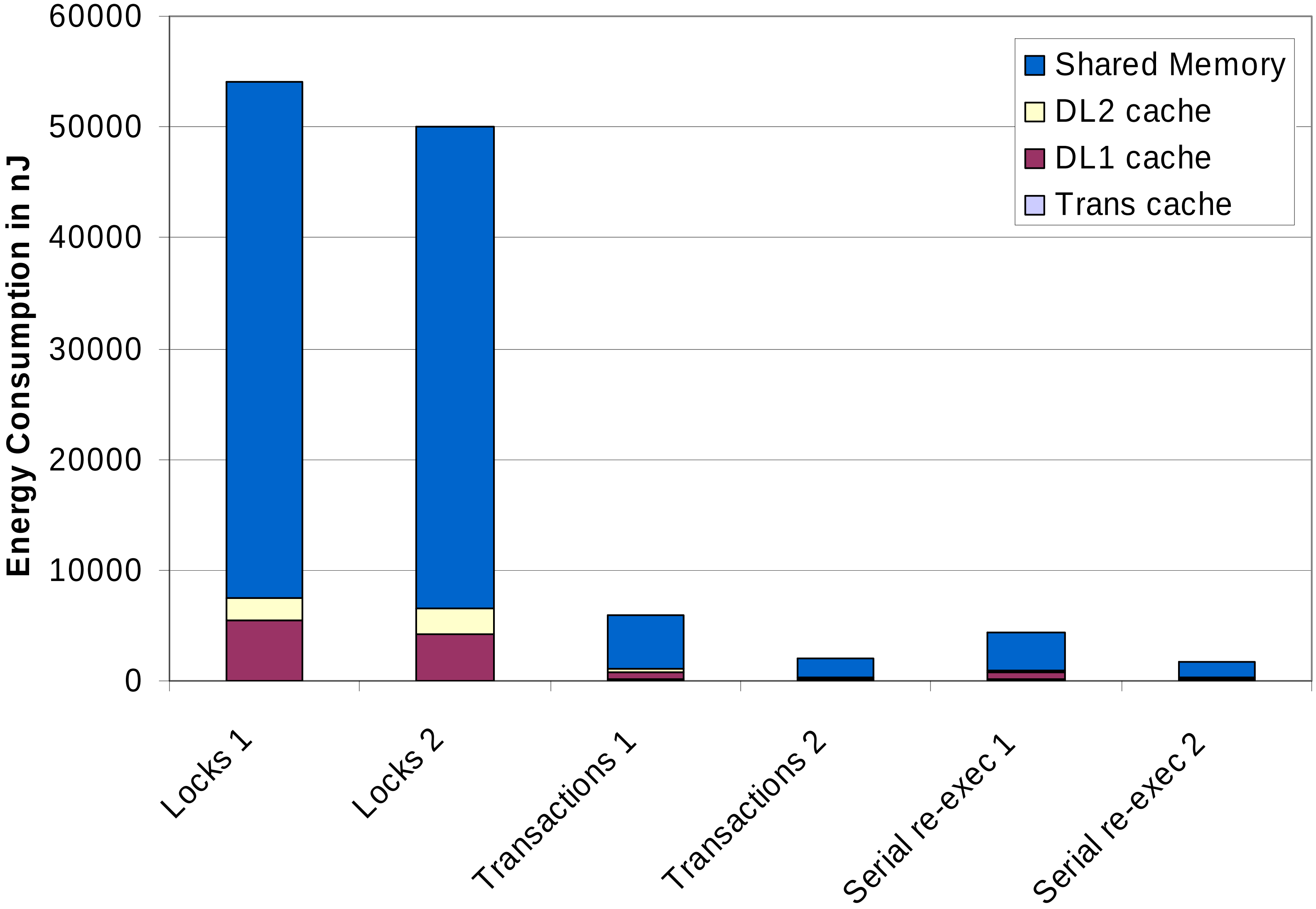
Standard Benchmarks Results



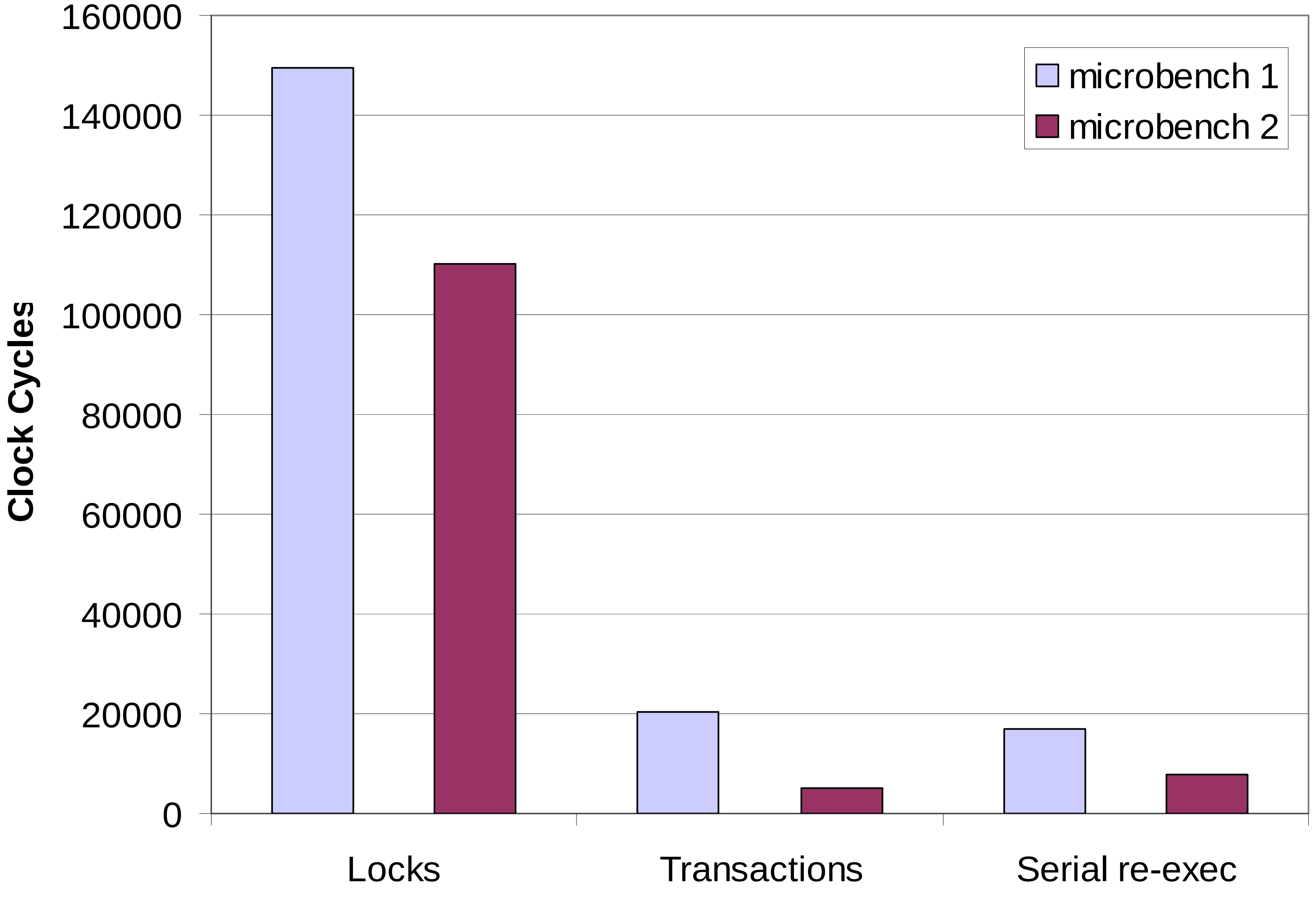
Synthetic Benchmarks

- Standard benchmarks have little contention
- Realistic applications include intervals of high contention
- Synthetic benchmarks
  - High contention
  - Various conflict scenarios
- Parallel accesses to a shared array

Energy Consumption
Locks vs. Transactions



Performance
Locks vs. Transactions



Conclusion

- Throughput and energy need to be balanced
- Speculative approach has a clear advantage in both energy and throughput in low contention
- Speculative approach needs modification in high contention for energy efficiency:
  → *serialized transactions*

Future Work

- Simulate a wider range of applications
- Various memory configurations
- Compare alternative locking schemes
- Consider longer running transactions
  - A trace-based analysis
  - Software transactions